DEFENCE R&D DÉFENSE

Enhancing TCP Performance over Satellite Channels

Maik Kammermann and Capt Hugues Latour
Defence Research Establishment Ottawa

Defence R&D Canada

DEFENCE RESEARCH ESTABLISHMENT OTTAWA

TECHNICAL REPORT
DREO TR 2000-079
November 2000

National Defence Défense nationale

Canada

Abstract

In general the Transmission Control Protocol (TCP) works well while establishing end-to-end connections for common Internet services. However, there are two problem areas associated with performance over satellite channels: propagation delay and channel noise effects.

Noise is a common problem in wireless technologies and consequently the bit-error-rate (BER) is significantly higher than in wired networks. TCP is particularly vulnerable to BER because it is a reliable protocol. TCP will retransmit lost or corrupted data when errors are detected. However, the main design goal was to avoid congestion-collapse in networks. The protocol has no means to decide whether a loss event was caused by congestion (buffer-overflow) or by corruption (noise, jamming). Nevertheless, TCP should react in a different way, depending on the type of errors: it should immediately retransmit outstanding data if the loss was caused by noise, and it should reduce network traffic if congestion was the reason for dropping data-packets.

Currently every loss indicates network congestion for the TCP standard version defined in RFC 0793. As a result it will reduce its sending rate significantly by invoking congestion control algorithms, regardless of the source of errors.

TCP is a "Sliding Window" protocol that uses acknowledgements to verify proper data delivery. Such protocols allow the sender to transmit only a given amount of data before receiving an acknowledgement in return. After each acknowledgement the window size is increased and a larger number of segments will be transmitted. The time TCP needs to reach maximum throughput is a function of the time needed for delivery and acknowledgement of data. It is obvious that long delays will hurt performance on links characterized by a large bandwidth-delay-product. In this case TCP will poorly utilize the available bandwidth. Furthermore, long delays increase the amount of time TCP spends to recover from losses while throughput is already very limited during these periods.

The Internet Engineering Task Force (IETF) recommended in RFC 2488 several mechanisms for a more efficient operation of TCP over satellite channels. The most important ones are called "Window Scaling" (RFC 1323) and "Selective Acknowledgements - SACK" (RFC 2018). The experiments described in this report were conducted to verify effects of these modifications. It was shown that using Scaling and

SACK improves performance on long delay paths. However, the link quality (BER<10^{-6}) is still an important foundation for providing good services.

Résumé

En général, le protocole TCP (Transmission Control Protocol) se comporte bien quand il s'agit d'établir une connexion de bout en bout pour des services Internet communs. Dans le cas de la performance sur des canaux de satellites, deux aspects causent problème : le délai de propagation et les effets du bruit.

Le bruit est un problème omniprésent dans les technologies sans fil et par conséquent leur taux d'erreur sur les binaires (paramètre BER) est beaucoup plus élevé que dans les réseaux câblés. Le protocole TCP est particulièrement vulnérable au taux d'erreur sur les binaires, puisqu'étant un réseau fiable il retransmet les données perdues ou corrompues quand des erreurs sont détectées. Cependant, le principal but de la conception est d'éviter les congestions-effondrements des réseaux. Le protocole ne dispose d'aucun moyen pour juger si une perte est due à une congestion (dépassement du tampon) ou à une corruption (friture, brouillage). Quoiqu'il en soit, TCP devrait réagir différemment, selon le type d'erreur; il devrait retransmettre immédiatement des données aberrantes si la perte est causée par du bruit, et il devrait réduire le trafic si la congestion explique pourquoi des paquets de données ont été rejetés.

Actuellement, chaque perte est interprétée comme une congestion du réseau pour la version standard de TCP, celle définie dans RFC 0793. Pour cette raison, il réduit le débit d'envoi considérablement en appelant les algorithmes de contrôle de la congestion, quelle que soit la source de l'erreur.

TCP est un protocole dit de « fenêtre mobile » qui fait usage d'accusés de réception pour vérifier le bon acheminement des données. De tels protocoles ne permettent la transmission que d'une certaine quantité de données avant la réception d'un accusé de réception. Après chaque accusé de réception, la taille de la fenêtre est augmentée et un plus grand nombre de segments seront transmis. Le temps nécessaire à TCP pour atteindre le débit maximal est fonction du temps nécessaire à la livraison et à l'accusé de réception des données. Il est évident que de longs retards freinent la performance sur les liaisons caractérisées par un grand produit largeur de bande-retard. Dans ce cas, TCP exploite maladroitement la largeur de bande disponible. De plus, de longs retards augmentent le temps que prend TCP pour la reprise après des pertes, justement lors de périodes où le débit est déjà très limité.

Le groupe de travail IETF (Internet Engineering Task Force) a recommandé dans RFC 2488 plusieurs mécanismes pour rendre plus efficace le fonctionnement de TCP sur des canaux par satellites. Les deux plus importants s'appellent « Échelonnement de la fenêtre » (RFC 1323) et « Accusés de réception sélectifs – SACK » (RFC 2018). Les expériences décrites dans le présent rapport ont été réalisées en vue de vérifier les effets de ces modifications. Elles ont démontré que l'échelonnement et le SACK améliorent la performance sur les trajets ayant de longs retards. Cependant, la qualité de la liaison (taux d'erreur sur les binaires $< 10^{-6}$) demeure toujours un préalable incontournable pour offrir de bons services.

Executive Summary

Background

Developed nearly 30 years ago the Transmission Control Protocol / Internet Protocol (TCP/IP) suite of protocols has become the most common communications platform today. Most Internet services rely on these protocols. Satellite links seem an ideal means for offering Internet services over long distances and to remote locations. Unfortunately, TCP and IP are not optimized for satellite conditions, which include large latency, high (asymmetric) bandwidth and high Bit-Error-Rates (BER). This environment constricts the performance of TCP in particular. Consequently, the throughput of TCP over satellite networks is usually restricted to only a fraction of the available bandwidth.

TCP's poor performance over satellite channels is caused by its internal flow control measures. It has been shown that long delays directly affect end-to-end performance because of TCP's acknowledgement algorithm. Furthermore, TCP responds to all losses by invoking algorithms that reduce the value of TCP's flow control parameter (TCP window) in an effort to avoid network congestion. As a result the transmission rate is dropped significantly because of a smaller TCP window. However, noise is today the more likely reason for losing data in wireless systems rather than congestion. Accordingly TCP should immediately retransmit missing data instead of reducing network traffic.

The Military Satellite Communications Group at the Defence Research Establishment Ottawa has been investigating TCP's specific problems in the satellite environment. Experiments have been conducted to evaluate mechanisms that have recently been recommended by the Internet Engineering Task Force (IETF) to enhance performance. Two such mechanisms involve selecting an appropriate TCP window size and invoking a selective acknowledgement scheme.

Results

The current TCP standard and optional additions were examined on links with significant latency. A network was established and experiments were conducted using different TCP window sizes and the benefits of selective acknowledgements. A data link simulator was first used as the transmission channel to generate latency and defined error-rates. The results were then verified by carrying tests over a SKYNET satellite.

It was shown that TCP's ability to establish high data rates on long delay circuits relies on an appropriate size of the TCP window. Bandwidth utilization by one single FTP-connection, established on a 1.544 Mbps link with BER < 10^{-7} and a latency of 560 ms which is typical in satellite environment, has become around 53% while using an appropriate value (128 kbit) for the flow control parameter and selective acknowledgements. Instead bandwidth utilization was just 25% when using the TCP standard under same conditions.

The limit for bandwidth utilization is attributable to TCP's Slow Start algorithm. This algorithm allows the TCP sender to transmit a limited amount of data, specified by the TCP window size, before receiving an acknowledgement from the receiver. If data transfer is successful, Slow Start will allow the sender to transmit a greater amount of data by increasing the TCP window size. The Slow Start algorithm is invoked at the beginning of a TCP connection as well as when transmission errors occur causing a timeout.

Managing flow control by determining the appropriate window size is complicated, because maximum throughput is a function of the entire end-to-end connection. The time for the transmission of data and acknowledging this data is very important. A significant latency has obviously bad influence on throughput especially during the Slow Start phase of TCP-based communications because it simply needs more time to reach an appropriate throughput or TCP window size.

Nevertheless, Slow Start is important for TCP's reliability. It provides the ability to adapt throughput to the actual network capacity. The factors responsible for the optimal window size are the bandwidth and the round-trip latency or delay found on a circuit. The bandwidth-delay product represents the maximum amount of data a network can handle. Setting a value that is larger has no positive effect but might cause recovery problems and waste resources. Conversely, a window will not lead to efficient network utilization if it is smaller than the bandwidth-delay product.

Significance

The importance of TCP/IP is increasing. Internet technology offers great opportunities and has already been introduced in many military communications and information systems.

Underlying networks often rely on satellite services but in this case TCP restricts performance.

Recommendations by the IETF on approaches to improve TCP performance have been implemented and verified. Using these optional additions to the current TCP standard provides better global communications capability and guarantees compatibility to existing systems.

Future Plans

This study has focused on the transport level of the OSI model. Future investigations may include the use of higher gain error correction codes to improve link quality at the data link level, or the use of multiple TCP connections for file transfers. As well, the performance of TCP/IP should not only be examined with geosynchronous (GEO) but could also include low-earth-orbit (LEO) and medium-earth-orbit (MEO) constellations that use intersatellitelinks and handovers for world-wide coverage which might cause further limitations.

Kammermann, M., Latour, H. (Capt), Enhancing TCP Performance over Satellite Channels, Defence Research Establishment Ottawa, DREO TR 2000-079, August 2000

Sommaire À L'intention De La Direction

Contexte

Créée il y a près de 30 ans, la suite de protocoles TCP/IP (Transmission Control Protocol/Internet Protocol) est de nos jours la plus utilisée des plates-formes de communication. La plupart des services Internet reposent sur ces protocoles. Les liaisons par satellites semblent être le moyen parfait pour offrir des services Internet sur de longues distances et à des localités éloignées. Malheureusement, TCP et IP ne sont pas optimisés pour les conditions de la transmission par satellite: grand temps de latence, importante largeur de bande (asymétrique) et taux d'erreur sur les binaires élevé. Tout particulièrement, cet environnement est un obstacle à la performance de TCP. Par conséquent, le débit de TCP dans des réseaux par satellites se situe généralement à une fraction seulement de la largeur de bande disponible.

La mauvaise performance de TCP sur les canaux de satellites s'explique par des mesures internes de contrôle du flux. Il a été démontré que de longs retards affectaient directement la performance de transmission de bout en bout en raison de l'algorithme d'accusé de réception de TCP. Au surplus, TCP réagit à toutes les pertes en appelant des algorithmes qui réduisent la valeur du paramètre de contrôle du flux de TCP (la fenêtre TCP), dans le but, bien légitime, de prévenir la congestion du réseau. Ce faisant, le taux de transmission chute significativement à cause de la diminution de la largeur de la fenêtre TCP. Nous savons aujourd'hui que c'est le bruit qui est le principal motif de la perte de données dans les systèmes sans fil et non la congestion. Pour toutes ces raisons, TCP devrait retransmettre immédiatement les données manquantes plutôt que de réduire le trafic du réseau.

Le Groupe des communications du satellite militaire du Centre de recherches pour la Défense – Ottawa a étudié des problèmes spécifiques au protocole TCP dans un réseau par satellites. Des expériences ont été réalisées pour évaluer des mécanismes recommandés récemment par le Groupe de travail technique sur Internet (IETF) visant à rehausser la performance. Deux de ces mécanismes agissent sur la sélection de la bonne largeur pour la fenêtre TCP et sur l'appel d'une solution d'accusé de réception sélectif.

Résultats

La norme actuelle de TCP et des compléments facultatifs ont été examinés sur des liaisons possédant des temps de latence considérables. On a établi un réseau, et des expériences ont été réalisées avec différentes dimensions de fenêtre TCP en exploitant les avantages des accusés de réception sélectifs. Un simulateur de liaison de données a d'abord servi à titre de voie de transmission en vue de générer un temps de latence et des taux d'erreurs définis. Les résultats ont été confirmés au moyen de tests avec un satellite SKYNET.

On a montré que la capacité de TCP à établir des débits de données élevés sur de longs circuits à retard dépend de la dimension appropriée de la fenêtre TCP. L'utilisation de la largeur de bande passante dans le cas d'une seule connexion FTP établie sur une liaison de 1,544 Mbit/s ayant un BER $< 10^{-7}$ et un temps de latence de 560 ms, ce qui est typique dans un environnement de réseau par satellites, est passée à 53 % quand on utilisait une valeur appropriée (128 kbits) comme paramètre de contrôle et des accusés de réception sélectifs. Par comparaison, cette utilisation n'est que de 25 % avec la norme TCP actuelle dans les mêmes conditions.

La limitation de l'utilisation de la largeur de bande est attribuable à un algorithme de début lent de TCP. Cet algorithme permet à un émetteur TCP de transmettre une quantité limitée de données, imposée par la dimension de la fenêtre TCP, avant de recevoir un accusé de réception de la part du récepteur. Si le transfert de données se déroule avec succès, le début lent autorisera l'émetteur à transmettre une plus grande quantité de données en élargissant la dimension de la fenêtre TCP. L'algorithme de début lent est appelé au début de la connexion TCP, de même que lorsque des erreurs de transmission provoquent une interruption.

La gestion du contrôle du flux au moyen de la détermination de la dimension d'une fenêtre est compliquée, car le débit maximal est fonction de la totalité de la connexion de bout en bout. La durée de la transmission des données et l'accusé de réception de ces données est très importante. Un temps de latence considérable a sans conteste un influence néfaste sur le débit, particulièrement durant la phase de début lent de la communication sur TCP, tout simplement parce qu'il faut plus de temps pour atteindre un débit ou une dimension de fenêtre TCP acceptables.

Malgré cela, le début lent est un composant important de la fiabilité de TCP. Il permet d'adapter le débit à la capacité réelle du réseau. Les facteurs responsables de la dimension optimale de la fenêtre sont la largeur de bande, le temps de latence aller-retour ou le retard déterminé pour un circuit. Le produit largeur de bande-retard est une indication de la quantité maximale de données que le réseau est en mesure de gérer. Imposer une valeur supérieure à ce produit n'a aucun effet positif, mais peut être à l'origine de problème de reprise et d'une perte de ressources. À l'inverse, une fenêtre trop petite que le produit largeur de bande-retard peut conduire à une utilisation inefficace du réseau.

Importance

Les protocoles TCP/IP deviennent de plus en plus importants. La technologie Internet est pleine de promesses et est déjà employée dans de nombreuses communications et systèmes d'information militaires. Les réseaux de base ont souvent recours à des services par satellites, mais dans ce cas TCP met un frein aux performances.

Les recommandations du Groupe IETF sur les méthodes d'améliorer la performance TCP ont été appliquées et vérifiées. L'utilisation de ces éléments facultatifs surajoutés à la norme TCP permet d'avoir un meilleur moyen de communicabilité mondiale et assure la compatibilité avec les systèmes existants.

Plans pour l'avenir

Dans les études envisagées, on pourra examiner l'emploi du codage pour corriger les erreurs de transmission sur des liaisons par satellites ou l'emploi de plusieurs liens TCP pour le transfert de fichiers.

La performance de TCP/IP ne devrait pas être évaluée seulement avec un satellite géosynchrone, mais devrait l'être aussi pour des constellations sur orbites basses terrestres et sur orbites moyennes terrestres, qui exploitent des liaisons intersatellites et des passages à une autre liaison pour obtenir une couverture mondiale, ce qui peut être la cause d'autres limitations.

Kammermann, M., Latour, H. (Capt), Amélioration de la Performance du TCP sur des Liens par Satellite, Le Centre de recherches pour la défense Ottawa, DREO TR 2000-079, août 2000. (en anglais)

Abbreviations

ACK	Acknowledgement
BER	Bit-Error-Rate
DARPA	Defense Advanced Research Projects Agency
FEC	Forward Error Correction
FTP	File Transfer Protocol
GEO	Geosynchronous
HDLC	High Data Link Control
HLEN	Header Length
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
k	Kilo (1024)
LAN	Local Area Network
LEO	Low Earth Orbit
M	Mega (1024^2)
MSS	Maximum Segment Size
MTU	Maximum Transport Unit
OSI	Open Systems Interconnection
PPP	Point-to-Point Protocol
RFC	Request for Comments (Internet Standard)
RTO	Retransmission Timeout
RTT	Round Trip Time
SACK	Selective Acknowledgements (RFC 2018)
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network

TABLE OF CONTENTS

Abstract iii

Résumé v

Executive Summary vii

Sommaire x

Abbreviations xiii

TABLE OF CONTENTS xv

Figures and Tables xvii

1 Introduction 1

1.1 INTRODUCTION TO TCP/IP 1
1.1.1 Internet Protocol *1*
1.1.2 Transmission Control Protocol *3*

2 Limitations of TCP/IP in the Satellite Environment 5

2.1 NOISE PROBLEM 5
2.2 DELAY PROBLEM 5
2.3 OTHER PROBLEMS 6

3 Bandwidth-Delay Product 7

3.1 CALCULATING THROUGHPUT 7
3.2 IETF RECOMMENDATIONS 8
3.2.1 Forward Error Correction *8*
3.2.2 Path Maximum-Transport-Unit Discovery *8*
3.2.3 Window Scaling *9*
3.2.4 Selective Acknowledgments (SACK) *9*
3.2.5 Timestamps *10*

4 Loss Recovery 11

4.1 CONGESTION CONTROL 11
4.1.1 Slow Start and Congestion Avoidance *11*
4.1.2 Fast Retransmit and Fast Recovery *12*
4.2 SELECTIVE ACKNOWLEDGEMENTS 13

5 Extending the Window size 15

5.1 LARGER INITIAL WINDOWS 15
5.2 WINDOW SCALING 15

6 Experimental Results 17

6.1 TEST ENVIRONMENT 17
6.1.1 Operating System *18*
6.1.2 Router Configuration *19*
6.1.3 Data Link Simulator *19*
6.2 BANDWIDTH UTILIZATION ON LONG DELAY PATHS 20
6.3 WINDOW SCALING AND BER INFLUENCE 21
6.3.1 Optimizing the Window Size *22*
6.3.2 Error Influence *23*
6.3.3 Conclusion on Using Window-Scaling and Error Impacts *23*
6.4 SACK AT DIFFERENT BERS 24
6.5 SKYNET EXPERIMENTS 25

7 Conclusions and Recommendations 29

8 References 31

ANNEX A: Tuning Operating Systems 33

ANNEX B: Router Configuration 35

ANNEX C: SKYNET Experiments (Results): 37

Figures and Tables

FIG.1.1: IP DATAGRAM FORMAT 2
FIG. 1.2: OSI AND TCP/IP LAYERING 2
FIG. 1.3: TCP SEGMENT FORMAT 3
FIG. 4.1: SLOW START 12
FIG. 6.1: "TCP/IP OVER SATELLITE"-TESTBED 17
TAB. 6.1: PARAMETERS DEFINING THE WINDOW SIZE IN LINUX 18
FIG. 6.2: LATENCY INFLUENCE ON BANDWIDTH UTILIZATION 20
FIG. 6.3: ENHANCEMENT BY SCALING AND SACK 21
FIG. 6.4: INFLUENCE OF SCALING AND BER ON THROUGHPUT (560MS DELAY CIRCUIT) 22
FIG. 6.5: SACK EFFICIENCY 25
FIG. 6.6:SKYNET-EXPERIMENT 26
FIG. 6.7: BANDWIDTH UTILIZATION ON SKYNET CHANNEL 27

1 Introduction

The Transport Control Protocol / Internet Protocol (TCP/IP) suite of protocols used for data transfer works with most transmission media. Satellite links seem an ideal means for offering TCP/IP based Internet services over long distances and to remote locations. Unfortunately, Internet protocols are not optimized for satellite conditions which include large latency, high (asymmetric) bandwidth and high Bit-Error-Rates (BER). This environment constricts the performance of TCP in particular. Consequently, the throughput of TCP over satellite networks can be restricted to only a fraction of the available bandwidth.

This document gives an introduction to TCP/IP and describes current standard algorithms that are responsible for the protocol's accomplishments and worldwide success.

TCP's specific problems in the satellite environment are explained and confirmed by experimental results. The Internet Engineering Task Force (IETF) has recently recommended several mechanisms to solve those problems [1]. Some of them will be introduced in this report. Experiments have been conducted to show actual results and these compare favorably with expectations.

1.1 Introduction to TCP/IP

TCP combined with IP is used in many network topologies and has become an accepted de-facto standard in Wide Area Networks (WANs). Its ability to accommodate a great variety of underlying technologies builds the foundation of the Internet. The original protocol family resulted from research funded by the Defense Advanced Research Projects Agency (DARPA) in the 1960s and 1970s, when this US-Agency first established standards for packet-switched-networking.

1.1.1 Internet Protocol

IP fulfills the duties of the network-layer in the Open Systems Interconnection (OSI) Model, see Fig. 1.1. By definition, IP provides no guarantee that one of its *datagrams* (IP's transfer unit composed of a IP-header [20-24 bytes[1]] and data) will be delivered.

[1] Byte is often replaced by the synonym octet when related to TCP/IP

But since IP is connectionless, each datagram, whose maximum possible size is 65,535 bytes, can be routed along different paths as it travels towards the destination.

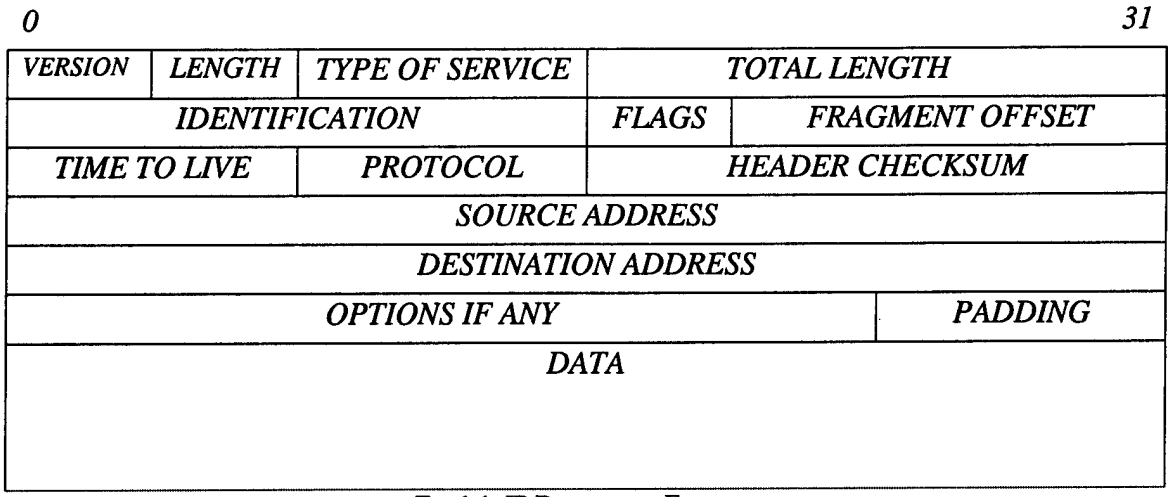

FIG.1.1: IP DATAGRAM FORMAT

Each routing process decrements the "Time-to-Live[2]" field in the IP-header by one. When this field reads zero, a router discards the datagram. This guarantees that if a datagram is not received within a certain "time" it will never be received. There is no possibility that datagrams get caught in routing loops forever.

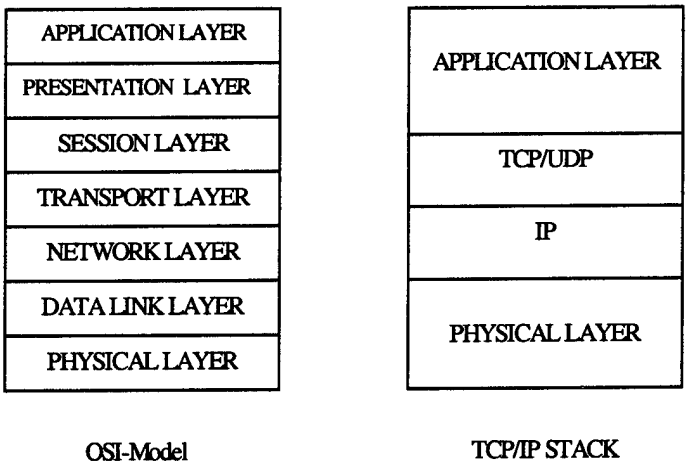

FIG. 1.2: OSI AND TCP/IP LAYERING

[2] In fact it is not a time but the number of routing processes allowed for data delivery

1.1.2 Transmission Control Protocol

TCP is implemented on top of IP to ensure reliable data delivery. As the name implies, TCP is a transport layer protocol. It uses a cumulative acknowledgement scheme and a "sliding-window" algorithm. TCP's data delivery unit is called a *segment.* The maximum segment-size (MSS) depends on the network[3]. Each segment is again divided into two parts as shown in Fig. 1.3, the TCP-header followed by data. A second form of transport layer protocol is the User Datagram Protocol (UDP). The main difference between TCP and UDP is that UDP does not guarantee reliable data delivery [2].

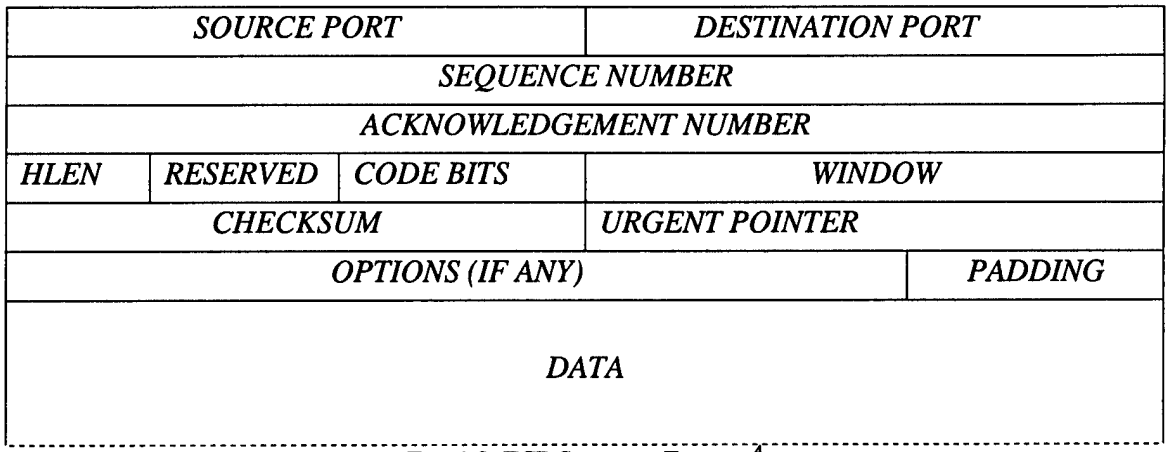

FIG. 1.3: TCP SEGMENT FORMAT[4]

When a TCP connection is established, the sender chooses a random 32-bit number as its own initial sequence number and informs the other node about its choice. Each subsequent byte transmitted by the node is assigned an incrementally increasing sequence number. A positive acknowledgement (ACK) containing the highest sequence number that has arrived is returned to the sender by the receiving node. This means ACK signals are not required for each byte because an ACK's sequence number indicates that all previous octets have been received successfully.

TCP uses a sliding window protocol. Such a protocol allows the sender to transmit a given number of segments before receiving an ACK. The maximum window size is limited to 64 kbytes in the TCP standard. When a connection is established TCP is

[3] Default value: 576 bytes; Ethernet: ca. 1500 bytes
[4] see [2] or [3] for detailed information about the TCP/IP format

initially very conservative on its use of network resources. The protocol uses an algorithm called Slow Start to increase the window to an appropriate size (see section 4.1.1).

In general TCP has two mechanisms for determining if segments are lost. The sender keeps a timer on each segment. If this timer reaches the Retransmission Timeout (RTO) value before a positive ACK signal is returned, the sender will automatically retransmit the segment. In addition, TCP uses the reception of duplicate ACKs as an indication that segments have been lost [2]. By default TCP assumes that any lost segments are due to network congestion. In reaction to the assumed network congestion, TCP will drop its transfer rate dramatically to avoid network collapse and buffer-overflow. After that TCP will try to increase its transfer rate again using the congestion control algorithms. These algorithms are more fully described in section 4.1.

2 Limitations of TCP/IP in the Satellite Environment

There are two main reasons for TCP's poor performance over satellite channels as compared to terrestrial links: noise and delay effects. These will be further discussed in the following subsections.

2.1 Noise Problem

Noise is a significant problem in wireless technologies. TCP has problems over wireless links because these channels exhibit a higher BER. This problem is aggravated in the military satellite environment because there is the potential risk of jamming attacks.

The TCP standard (RFC 0793) has no means to decide whether a loss was caused by corruption or by congestion. However, the action that TCP should take in the two cases is entirely different so differentiation is particularly important. In the case of corruption, TCP should merely retransmit the damaged segment as soon as a loss is detected. On the other hand, when the sender detects congestion, TCP should reduce network traffic. By definition TCP assumes every dropped packet is an indicator of network congestion. Its algorithms will reduce the window size and thus the transfer rate in an attempt to alleviate the congestion. This will not solve the problem of errors introduced by noise, but will decrease the throughput significantly.

Engineers are striving to develop better ways to address the noise issues. One method has been to minimize the noise by minimizing the signal bandwidth. The less spectrum space a signal occupies, the less noise passes through the receiving circuitry. However, this approach limits the maximum speed at which messages can be delivered. Another solution to the corruption problems is to improve the quality of wireless links by adding redundancy codes like Forward Error Correction (FEC) to the datastream. However, this increases signal bandwidth and processing complexity.

2.2 Delay Problem

It was previously mentioned that because of TCP's sliding window mechanism, the sender is allowed to transmit only a given number of segments before receiving an ACK. Within an ACK the receiver advertises a window size according to its available buffer. The sender will not transmit more data than advertised before receiving the next ACK

carrying the next window announcement and will transmit even less while establishing a connection or recovering from loss events.

It is obvious that a long delay between sending data and receiving an ACK has a detrimental effect on throughput. Long delays are normal in the satellite environment. For example, due to the speed of light, the typical round trip delay for links that include a satellite in geosynchronous orbit (GEO, altitude ≈ 36,000km) is 540-560ms. In some satellite environments, such as low earth orbit (LEO) constellations, the propagation delay is shorter but the delay varies over time. It is an interesting question whether this will have an impact on TCP performance. Furthermore, the design of many LEO systems includes handovers between satellites. It is not unlikely that this will be a source for additional packet delays.

2.3 Other Problems

Other limitations of TCP may occur in satellite networks that exhibit bandwidth asymmetry, with a larger data rate in one direction than the reverse direction because of limits on transmission power and antenna size at one end of the link. Some other systems are unidirectional using a non-satellite return path (e.g. a dialup modem link). The nature of most TCP traffic is asymmetric with data flow in one direction and ACKs in return. However, in this case the term asymmetric refers to different physical capacities in forward and return channels. For example, the reverse channel in many VSAT networks is shared among multiple satellite receivers. This can already result in a severe level of asymmetry, which might impact TCP performance.

3 Bandwidth-Delay Product

A very significant parameter for TCP's performance problem on links with high bandwidth and long round-trip delays is the product of both of them. The bandwidth-delay product defines the number of bits it takes to "fill the pipe", i.e., the amount of unacknowledged data that TCP must handle in order to make steady use of the available bandwidth. Performance problems arise when this product exceeds 10^5 bits. High-capacity packet satellite channels are of that type. For example, a T1-speed satellite channel (1.5 Mbps) has a bandwidth-delay product of nearly 10^6 bits. Proposed terrestrial fiber-optic paths will also fall into this category.

The reasons for poor performance on connections with large bandwidth-delay products are given in the traditional TCP standard specification itself and are further expanded on in the following sub-sections. However, some extensions have already been suggested to make better utilization of bandwidth. Five of these IETF recommendations are presented in section 3.2.

3.1 Calculating Throughput

TCP's receiving window size is particularly important because throughput is bounded by the Round Trip Time (RTT). Throughput is limited to:

$$Throughput_{max} = \frac{TCP\ window}{RTT} \qquad (1)$$

With the standard window size of 64 kbyte maximum throughput over a GEO satellite becomes:

$$Throughput_{max} = \frac{64kbyte}{560ms} \qquad (2)$$

$$\approx 112{,}000\frac{byte}{s} = 896{,}000\frac{Bit}{s}$$

3.2 IETF Recommendations

In designing new implementations of TCP we must pay careful attention to interoperability with the existing standard. To ensure interoperability the IETF recommends that TCP confirms applicability in "initial options", i.e., it may appear in the first segment[5] while establishing the connection. It is likely that most implementations will properly ignore any options in the first segment that they do not understand, so initial options will not cause a problem. On the other hand, receiving unexpected non-initial options may cause some TCP's to crash.

Therefore, in each of the extensions that have been proposed by the IETF, non-initial options may be sent only if an exchange of initial options has indicated that both sides understand the extension. This approach will also allow a TCP to determine the size of the TCP header it will be using.

3.2.1 Forward Error Correction

The first approach to improving the operation of TCP is not a TCP option, but the use of Forward Error Correction (FEC). TCP is a reliable protocol, meaning that lost or corrupted data is retransmitted when detected therefore unnecessary packet drops have to be avoided. Using FEC will help to make the link as "clean" as possible. However, it will increase overhead. Typically most satellite communication links already employ FEC. Using FECs with higher coding gain should increase the efficiency of TCP but at an increase in computational complexity.

3.2.2 Path Maximum-Transport-Unit Discovery

Path Maximum-Transport-Unit (MTU) Discovery (RFC 1191) allows TCP to use the largest possible segment size in a network , meaning more data bytes per overhead will be sent. Normally the TCP sender transmits data appropriate for its local network through a priori knowledge. However, if the receiver is not on the local network the segment size selected may be too large. Increasing TCP's window is based on the segment size and therefore a larger MTU will enable the TCP sender to increase the window, in terms of bytes, more rapidly. However, any intervening gateway will fragment the packets when they are too large to be forwarded through a following network. At the destination the receiver is responsible for the reassembly with an increase in computational overhead. In addition, a large MTU is affected by poor BERs due to a correspondingly higher

[5] First segment with Sync-flag = 1.

probability of error within any given segment. Because of that more retransmissions might be triggered when using large segments in a "fragile" network.
If Path MTU Discovery is used the sender sets the "Do not fragment (DF)" bit in the TCP header. In the case that a segment is too large for a gateway to forward without fragmenting, an Internet Control Message Protocol (ICMP) message, indicating that the packet is too large to be forwarded, is returned to the sender including information about the largest possible segment size. The sender will choose an appropriate size and retransmit the segment. This process continues until the segment finally reaches the receiver. Path MTU Discovery will cause a delay before TCP is able to send. After that no additional time and buffer are needed for fragmentation and reassembly. In practice it does not consume a great amount of time to adjust the MTU of a circuit due to the support of common values.

3.2.3 Window Scaling

The standard TCP header (Fig. 1.3) uses a 16-bit field to report the receive window size (receive-buffer) to the sender, therefore the largest window that can be used is $2^{16} = 64$ kbytes[6]. To circumvent this problem a TCP option has been created to allow windows larger than 2^{16}. This option, "Window Scaling" (RFC 1323), defines an implicit scale factor, to be used to multiply the window size value that can be found in the TCP header. This will allow TCP "to keep more packets in flight". See section 5.2 or [4] for more information.

3.2.4 Selective Acknowledgments (SACK)

Packet loss can have a catastrophic effect on throughput. This effect is exaggerated by the simple cumulative acknowledgment scheme of TCP. Whenever segments are lost, the transmitting TCP will slow down and begin retransmission. Using only standard congestion control algorithms the sender might be forced to retransmit segments that have already been received and queued by the receiver, which is not efficient.
By using an option called "Selective Acknowledgments" (SACK), the receiver can inform the sender about all the segments that have arrived successfully to avoid unnecessary retransmission. SACK is described in section 4.2. For the definition in detail see RFC 2018 [5].

[6] Some TCP implementations only use 15 bit (LINUX) or less (WINDOWS NT, 14 bit)

3.2.5 Timestamps

TCP implements reliable data delivery by measuring the RTT, the time interval between sending a segment and receiving an acknowledgment for it. Experience has shown that accurate, current RTT estimates are necessary to adapt to fast changing traffic conditions. Without an accurate RTT estimate, a busy network is subject to an instability known as "congestion collapse".

Measuring a segment's RTT may involve a non-trivial amount of computation in some implementations. This is due in part because TCP segments may be repacketized upon retransmission, and in part because of complications due to the cumulative TCP acknowledgement, To minimize this computation, some implementations time only one segment per window. While this yields an adequate approximation to the RTT for small windows (e.g., a 4 to 8 segment Arpanet window), for a high-bandwidth connection it results in an unacceptably poor RTT estimate.

In the presence of errors, the problem becomes worse. It is not possible to accumulate reliable RTT estimates if retransmitted segments are included in the estimate. Since a full window of data will have been transmitted prior to a retransmission, all of the segments in that window will have to be acknowledged before the next RTT sample can be taken. This means at least an additional window's worth of time between RTT measurements and, as the error rate approaches one per window of data (e.g., 10^{-6} errors per bit), it becomes effectively impossible to obtain an RTT measurement. A TCP "echo" option has been proposed that enables each segment to carry its own timestamp (RFC 1323). This will allow every segment, including retransmissions, to be timed at negligible computational cost.

4 Loss Recovery

To understand the problems on links with a large bandwidth-delay product, and the meaning of SACK, it is necessary to have a closer look at TCP's congestion control algorithms and retransmission strategy [6]. This section will introduce the standard algorithms and the special meaning of SACK.

4.1 Congestion Control

Today's TCP standard contains algorithms to keep networks from suffering congestion. These procedures are called Congestion Avoidance, Fast Retransmit, Fast Recovery and even Slow Start. A short description of these procedures is given in the following section. See RFC 2001 for even more details.

4.1.1 Slow Start and Congestion Avoidance

Slow Start is the procedure that is used to establish a connection or to restart a connection after RTO occurs. Its purpose is to ensure that the data throughput is appropriate for the network.

Slow Start functions by gradually increasing the rate at which the sender injects data into the network. In this algorithm the sender uses a variable called "Congestion Window (cwnd)". The value of cwnd is the maximum number of data segments that have been sent but not yet acknowledged. The procedure starts by sending just one segment (cwnd=1) and waiting for an ACK. For each ACK the sender receives, cwnd will be increased by one segment. In effect, two new segments will be transmitted after an ACK has reached the sender, leading to an exponential growth of the amount of data (Fig. 4.1).

The sender can transmit up to the lesser of cwnd or the receiver's advertised window. Therefore, cwnd is flow control imposed by the sender while the advertised window is flow control imposed by the receiver. The amount of time needed for cwnd to reach the advertised window size can be estimated by the following equation:

$$\textit{Slow Start Time} \approx RTT \cdot \log_2(\textit{advertised TCP window size in segments}) \qquad (3)$$

Fig. 4.1 shows that the amount of data injected into the network is doubled almost every RTT. This might be too much when cwnd reaches a certain value. If this is the case, then the Congestion Avoidance algorithm will replace Slow Start and increase cwnd by only one additional segment within the estimated round trip time. This algorithm leads to only linear growth of cwnd. However, the limit for cwnd is always the receiver's advertised window.

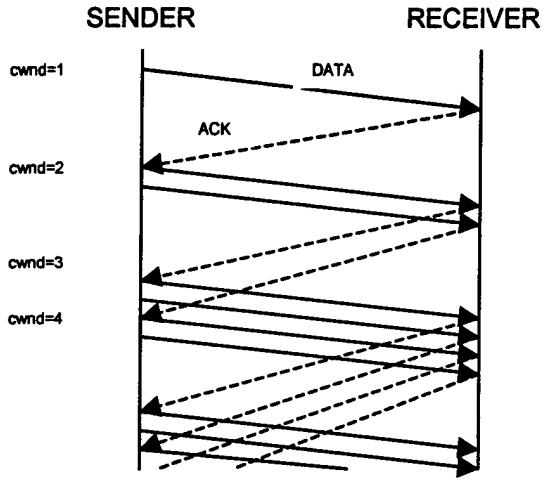

FIG. 4.1: SLOW START

Since Slow Start and Congestion Avoidance are two independent algorithms with different objectives another variable is required when they are implemented together. The Slow Start Threshold (ssthresh) will set the cwnd-limit for Slow Start and the starting point for Congestion Avoidance. For the combined algorithm see [6].

Because the amount of time required for Slow Start and Congestion Avoidance to achieve full bandwidth is a function of RTT, satellite links are particularly sensitive to the limited throughput available during Slow Start and Congestion Avoidance [7].

4.1.2 Fast Retransmit and Fast Recovery

Fast Retransmit reduces the time it takes a TCP sender to detect a single dropped segment. Rather than waiting for the RTO, the TCP sender can start retransmission if it receives duplicate (usually three) identical ACKs. The missing segment will be sent

immediately while the receiver is buffering subsequent segments. Successful retransmission is indicated by an ACK one RTT later. This ACK will include the segments that have been successfully received and queued in the meantime.

Fast Recovery works hand in hand with Fast Retransmit. As mentioned above, when a sender retransmits a segment, it normally recovers by moving first into Slow Start followed by Congestion Avoidance. If the sending TCP detects the segment loss by receiving duplicate ACKs, Fast Recovery is used instead. Fast Recovery will set ssthresh to half the current value of cwnd when the loss was detected. It thereby skips a Slow Start phase and triggers Congestion Avoidance. See [6] and [8] for a detailed description.

As a result, a loss detected by RTO causes a Slow Start phase while loss detection by duplicate ACKs begins Congestion Avoidance immediately without falling back into Slow Start which is the primary factor limiting TCP's performance.
The time until RTO occurs is not a fixed value. It is calculated by the current RTT. So the steady measurement of RTT becomes a key issue [3].

4.2 Selective Acknowledgements

Packet loss in a network can reduce throughput drastically. If only one segment is missing, Fast Retransmit and Fast Recovery will be in charge to keep the connection alive. But even with these algorithms, the loss of more than one packet within a window can not be handled and it is quite likely that a RTO will occur. The Slow Start procedure will get things going but it will take a few RTTs to regain performance.
Selective Acknowledgements (SACK) were proposed to handle more than one dropped packet. TCP with SACK is specified in RFC 2018 [5]. Its design goal is to provide information about non-contiguous blocks of data that were received. The receiver uses SACK to inform the sender of all segments that were transferred successfully but have not been cumulatively acknowledged. The information is stored in the TCP header option field. The ACK field itself will not be changed and so Fast Retransmit/Fast Recovery will be triggered by duplicate ACKs. The SACK information is then used for retransmission of only those lost segments within the next window of data probably before a RTO occurs. In place of pure Fast Retransmit/Fast Recovery algorithms, the combined algorithms and SACK should handle more than one segment loss per window of data.

A Sender and receiver must both run a version of TCP with SACK to use this option. Compatibility to non-SACK versions is achieved by the initialization handshake. The SACK announcement is made at the initialization of a connection in the first segment with the SYNC flag set. The receiver must send its own SACK-permitted option in its answer. If sender and receiver agree about using SACK, it will be used during the connection. This option would not be used for the connection if there was not a SACK-permitted option in either the sender's or receiver's first SYNC segment. So SACK and Non-SACK TCP versions can still work together.

While exchanging data a receiver will use the SACK option in its header to inform the sender of non-contiguous blocks of data that have been queued in the receive buffer within the immediate window size. The SACK option will be present in each segment in which the ACK flag that does not acknowledge the highest sequence number received. Data already received will be given in blocks to allow the sender to retransmit only the missing segments in the Fast Retransmit and Fast Recovery phases of TCP. Depending on the options used for the connection one, two or three SACK blocks may fit into a segment.

5 Extending the Window size

TCP's window size has great influence on throughput as defined earlier. Larger windows can enhance performance on links with a high bandwidth-delay product. Unfortunately the probability of errors within a larger window will also increase. The selection of the appropriate window size is not trivial and must be done with care.

5.1 Larger Initial Windows

TCP implementations use Slow Start in as many as three different ways: (1) to start a new connection, (2) to restart a transmission after a long idle period, (3) to restart after timeout (RTO). The initial Slow Start window was specified in [6] to one segment. A larger initial window up to 4kByte is proposed in [9] to allow higher throughput during first RTTs. This proposal has already been shown to improve performance of TCP connections over satellite channels, but it is still in an experimental stage.

5.2 Window Scaling

A 16-bit field in the TCP-header limits the maximum window size to 64 kbytes. Changing the header would result in losing compatibility to older TCP implementations. An IETF recommendation to use window scaling has been proposed to solve this problem using one of the headers option fields.

To implement a TCP option that allows windows of extended size by scaling, the sender must reliably know the receiver's current buffer-size or scale factor. A very simple approach using an ACK to provide this information to the sender will not work, because acknowledgement segments will not be delivered reliably (unless the ACK happens to be piggy-packed[7] on data). However, SYNC segments are always sent reliably, suggesting that each side may communicate its window scale factor in an initial TCP option. This approach has a disadvantage; the scale factor must be established when the connection is opened, and cannot be changed thereafter. Nevertheless, other alternatives would be much more complicated.

"Window Scaling" was proposed in RFC 1323 [4]. It is a three-byte option that may be in the SYNC segment. The option communicates a scale factor that serves a dual purpose

[7] Data segment would carry ACK if dataflow was bi-directional

because it signals that TCP is ready to both send and receive larger windows of data. Both sides must send the scaling option to be enabled. The scale factor is logarithmically encoded as an integer power of two, enabling easy implementation with binary shift operators.

In order to preserve the integrity of new segments in the network, the window size must not exceed 2^{30}. Otherwise new data would be considered old and discarded by the receiver. If the standard TCP window is 65535 bytes (2^{16}-1), the maximum scale factor is 14. This allows a 1 Gbyte window. See [4] for details.

6 Experimental Results

Several mechanisms have been introduced that should aid TCP performance in a satellite environment. Two of the proposals, which have the potential for the most benefit were experimentally examined: Window Scaling (RFC 1323) and Selective Acknowledgements (2018). The experiments reported in this paper were conducted to show performance gains in actual scenarios.

6.1 Test Environment

To quantify the performance of TCP over long delay paths a separate network was built for test purposes and is shown in Fig. 6.1. Two IP-subnets[8] represented by Pentium-II class machines were connected through routers (CISCO 1600) and a data link simulator (ADTECH SX/12). The simulator has provided flexibility in setting link parameters for bandwidth, delay and errors.

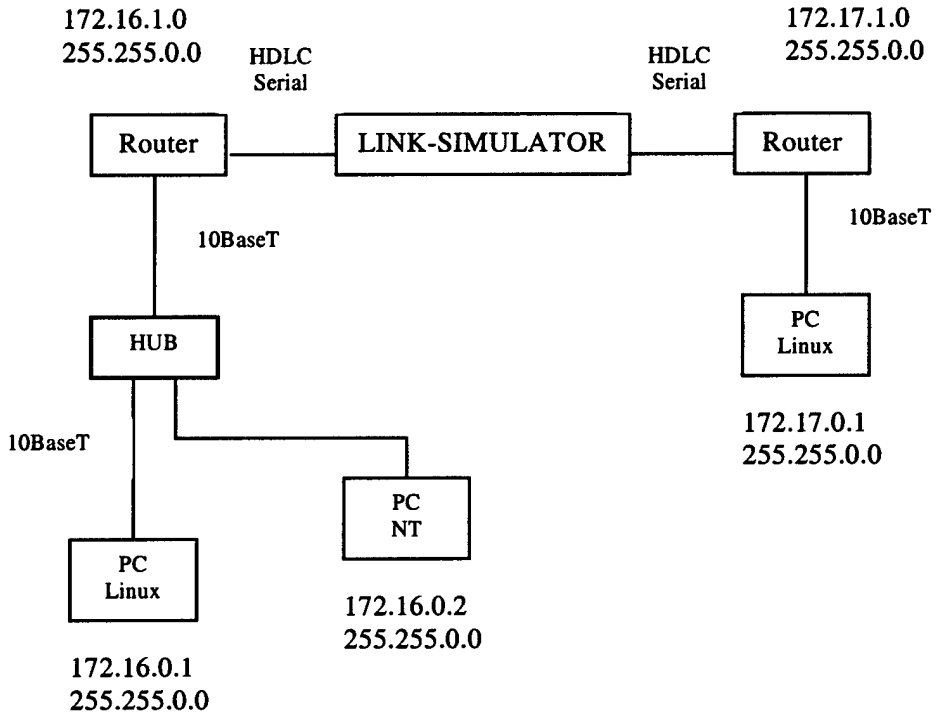

FIG. 6.1: "TCP/IP OVER SATELLITE"-TESTBED

[8] 172.x.x.x addresses were used, although there was no connection to other networks

6.1.1 Operating System

Linux (Version: 2.2.-15) was chosen as the operating system for the PCs representing TCP sender and receiver. It is important to know that Linux uses only 15 bit (32 kbyte) of the window size header field by default. But this is more than the WindowsNT 4.0 or Solaris 2.6 default of 8 kbyte. In addition, Linux supports the IETF recommendations mentioned above [10].

Although Linux does not use the whole TCP window field by default, it was the optimal choice as operating system because it allows the tuning of a running system. Changing the parameters in table 6.1 will change the values in the kernel without the need to reboot the machine. The results of the changes can easily be monitored and verified by using a network analyzer (LAN-Watch) to examine the data. When observing transmissions the window announcement will change dynamically during data-transfer depending on the available receive-buffer, however the TCP sender and receiver will agree on using Window Scaling or SACK or both when a port-to-port connection is established. Therefore, the first segment containing a SYNC-signal is always of particular interest. It should be noted that a standard FTP-transfer will open at least three ports: Authentication, FTP-Control (port 21) and FTP-Data (port 20).

To tune the default and maximum values simply edit the files listed in table 6.1, change and save the values[9]:

Default receive window	*/proc/sys/net/core/rmem_default*
Maximum receive window	*/proc/sys/net/core/rmem_max*
Default send window	*/proc/sys/net/core/wmem_default*
Maximum send window	*/proc/sys/net/core/wmem_max*

TAB. 6.1: PARAMETERS DEFINING THE WINDOW SIZE IN LINUX

Linux also offers the opportunity to switch TCP options on or off. The parameters can be found in /proc/sys/net/ipv4/. Any editor can be used to decide whether to use tcp_timestamps, tcp_windowscaling, tcp_sack, etc. by changing the parameter's value to 1 (enabled) or 0 (disabled). Note that by default the options mentioned above are enabled. Because of the flexibility Linux provides it is not necessary to reboot or even rebuild the kernel.

[9] Even if you cannot read the default values with your editor (gedit) changes are accepted

All the above values are set and used by header files in the Linux kernel source directory */usr/src/linux/include/linux/skbuff.h, /usr/src/linux/include/net/tcp.h* and *sock.h.* Modifying those files and recompiling the kernel should result in persistent changes (see Annex A).

6.1.2 Router Configuration

Routers work at the network layer (OSI Layer-3). In general routers can connect hosts in different networks by using logical addresses instead of the hardware addresses. IP is the most common network layer protocol. Every host in an IP network can be identified because of its 4-byte IP-address. In addition, the IP-address concept includes the opportunity to build subnets. Two IP-subnets were connected for the "TCP/IP over Satellite" testbed through a pair of Cisco routers (1600 Series). To configure the routers, IP-addresses (subnets) have to be assigned to the routers' network interfaces[10]. Annex B shows the actual router configuration which has been used for the experiments. It was built manually by using the related terminal-commands described in [10].

The configuration routes TCP/IP services between the two subnets (172.16.x.x and 172.17.x.x) for static routing over a synchronous serial line. The default encapsulation type used on leased lines between two Cisco routers is the High Data Link Control (HDLC) but it was shown that Point-to-Point Protocol (PPP[11]) encapsulation worked without loss of performance.

There were no problems attributed to the routers during any experiment. Nevertheless, it might be possible to gain performance by changing the routers standard queuing strategy.

6.1.3 Data Link Simulator

Channel parameters for the experiments were set with the Adtech SX/12 Data Link Simulator. To simulate a satellite T1-channel the data rate was set to 1,544 kbps and the main delay parameter was set to 280ms for both channels – or 560ms for a return trip delay. The random error rates were set equally for both directions. Only the random error mode was used during the experiments. It was assumed that burst errors would be mitigated using interleaver techniques and so the random model is assumed valid.

[10] Cisco 1600: Ethernet0=LAN, Serial 0=WAN; Ethernet0 and Serial0 share one IP-address
[11] PPP must be used to connect routers from other vendors

The extended T1/E1 Simulation Option has not been used in this experiment but it could be used in future work [12].

6.2 Bandwidth Utilization on Long Delay Paths

To demonstrate the problems around long delay paths the bandwidth utilization was estimated by means of FTP transfers. The Linux TCP default parameters were used (effective Window 32 kbyte). The time was measured for transfers of 1 Mbyte files, the average speed and bandwidth utilization were calculated for different delay parameters.

The experimental results shown in Fig. 6.2 underline the performance problems of TCP on paths with large bandwidth-delay products. With a delay of 560ms, even without errors, the bandwidth utilization became less than 25% on a 1,544kbps link. The values beside the data points represent the actual bandwidth utilization (1,544 kbps = 100%) according to the current delay set with the data link simulator.

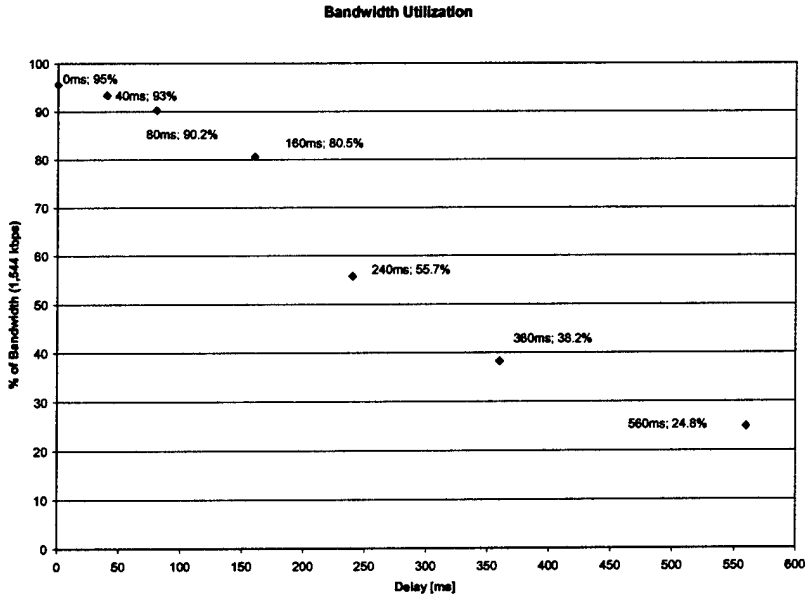

FIG. 6.2: LATENCY INFLUENCE ON BANDWIDTH UTILIZATION

Fig. 6.3 gives the experimental results when using Window Scaling and SACK for various time delays over a 1,544 kbps link. A file of 1 Mbyte was transferred 100 times. The time was measured for each FTP-Transfer and the average was calculated. Measurements were made with delays from 0 to 600ms in increments of 100ms. The

chart shows the results for the Linux standard window (32 kbyte effective) with SACK disabled and for a window of double size (64 kbyte effective) with SACK enabled. No errors were injected. The positive influence of a window of double size (64 kbyte) on throughput is quite obvious.

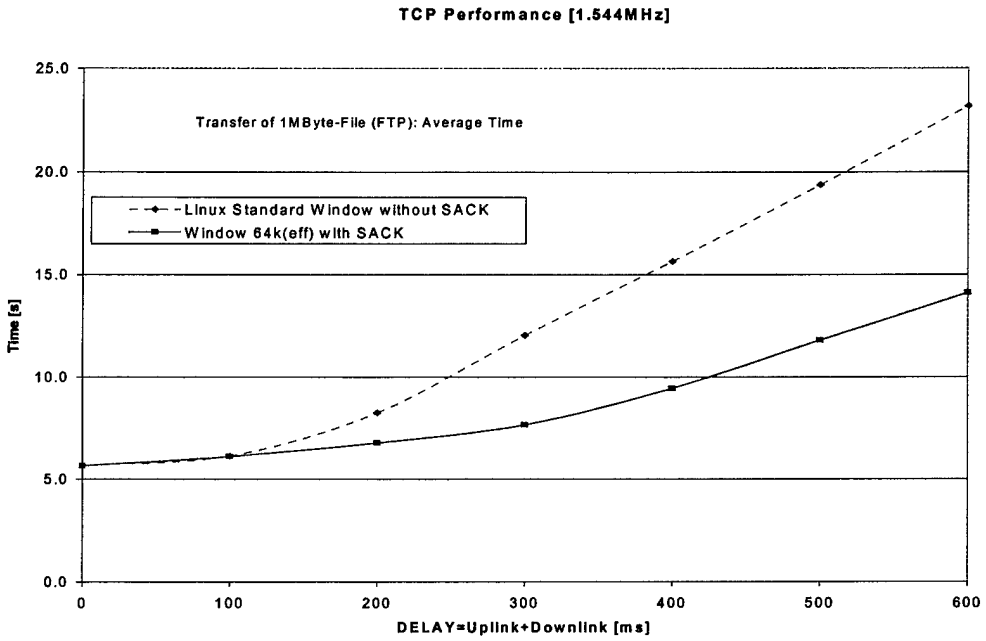

FIG. 6.3: ENHANCEMENT BY SCALING AND SACK

6.3 Window Scaling and BER Influence

After Slow Start has fully opened the initial congestion window, TCP's flow control services depend upon the receiver's advertised window. This value dictates the maximum amount of data that can be sent during steady-state operations without the sender having to stop and wait for an ACK. As such, the receiver's window is central to achieving efficient throughput because it determines the smooth flow of data more than any other element. Most applications relying on TCP use the system-wide default, which is sub-optimal. Therefore, one way to improve performance for any given system is to optimize the default size of the receive window. The bandwidth-delay product of a circuit can serve as estimation for an appropriate value.

6.3.1 Optimizing the Window Size

A connection of 1,544 kbps was established for the experiments. The delay was set to 560ms. The window size was varied; SACK was always enabled. A 1 Mbyte file was transferred 100 times for each set of parameters including different link qualities. The average values for transmitting the file at various window sizes are shown in the Fig. 6.4.

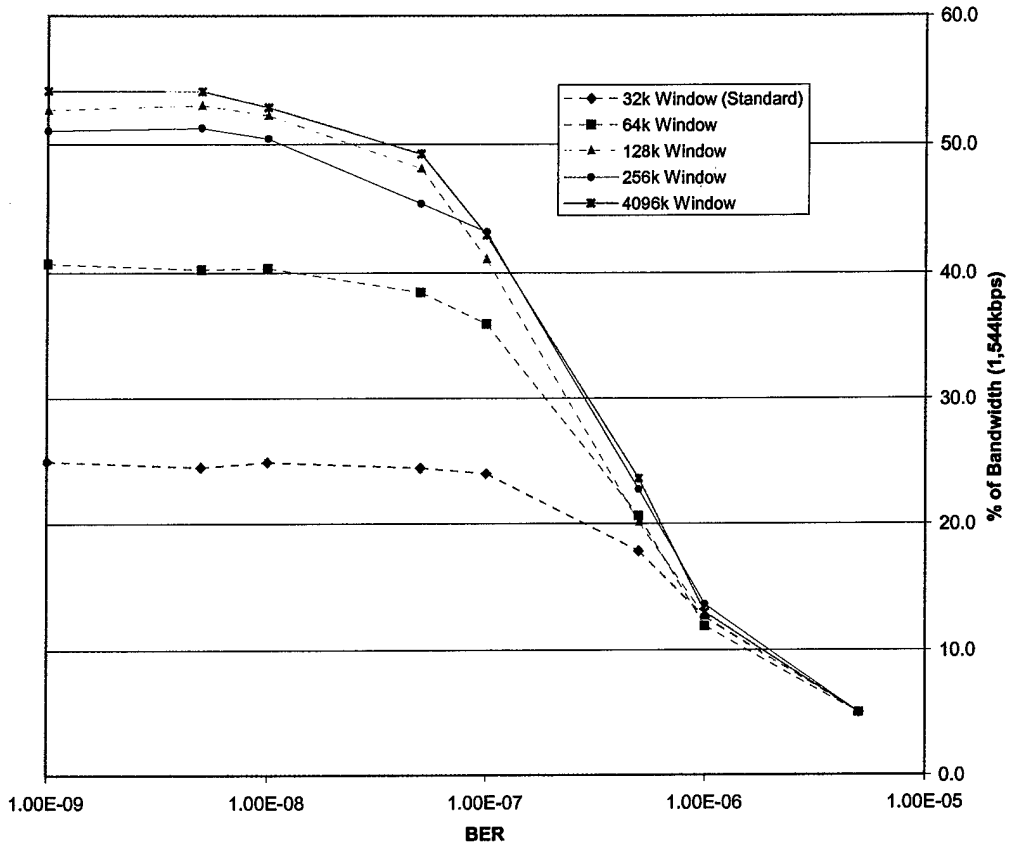

FIG. 6.4: INFLUENCE OF SCALING AND BER ON THROUGHPUT (560MS DELAY CIRCUIT)

The experimental results shown in Fig. 6.4 underline that even on a link of high quality (BER<10^{-7}) bandwidth utilization is only 25% of 1,544kbps for an effective window of 32 kbyte. Doubling the window (64 kbyte) leads to a utilization of 40%, doubling again to 128 kbyte leads to a utilization of 53%. Increasing the window size above 128 kbyte had minimal effect on performance. Even with a window of 4,096 kbyte only 55% of the bandwidth were used by the established point-to-point connection.

Higher figures have probably not been reached because every TCP connection begins with Slow Start. The time needed for increasing the window depends on RTT which is very high on satellite links. According to a calculation made in [13] for a 1,544 kbps link over GEO-satellites around 5.6s are required to open a window size of 64 kbyte. As a result, much of the data is already transferred within smaller windows even if the maximum size is large. In conclusion, throughput figures are heavily influenced by Slow Start. It is anticipated that larger windows will have minimal positive effect when transferring only a few kbytes.

6.3.2 Error Influence

Bandwidth and latency are not the only things that affect throughput as shown in Fig. 6.4 In general, every error slows down data transfer and forces at least Fast Retransmit/Fast Recovery algorithms to take over and reduce the current window size. With the same error probability more errors occur within larger windows. Because of that the large window never becomes really efficient.

It can be seen that performance rapidly degrades when the BER becomes worse than 10^{-7}. Independent of the window size, bandwidth utilization has been measured 12-14% at BER=10^{-6} and just 5% at BER=$5*10^{-6}$.

6.3.3 Conclusion on Using Window-Scaling and Error Impacts

Setting the window size accurately is absolutely crucial to achieving optimal performance over long delay paths. The experimental results have underlined that setting the receiver buffer too small results in an artificial bottleneck, where the window is smaller than the amount of data that the network can handle. In this model, the sender has to wait for the next ACK before it starts sending again even while the network may be idle having plenty of capacity left.

Conversely, setting the window substantially larger than the bandwidth-delay product has not enhanced performance. It may result in a waste of system resources instead because some systems allocate memory according to the size of receive buffers. But the main risk is obviously the steady attempt to put more data into the pipe than the network can handle. This can easily lead to data loss in complex networks if the buffering at gateways is not appropriate.

The most important requirement to achieve acceptable performance and bandwidth utilization is a link of high quality with small BER. Otherwise, TCP's congestion control algorithms will slow down data transfer even when using larger windows.

6.4 SACK at Different BERs

The contribution of SACK to enhance performance as shown in Fig. 6.4 has been difficult to measure. Remembering that SACK's role is to handle more than one error in a window of data and to avoid RTO, the influence of SACK should increase with the window size because this scenario becomes more likely. Nevertheless, duplicate ACKs will trigger Fast Retransmit/Fast Recovery algorithms after the first error. The benefits provided by SACK can only be recognized if at least a second error occurs during the recovery phase.

Fig. 6.5 shows average values for 1 Mbyte FTP transfers with and without SACK option in random-error scenarios. The enhancement achieved by the SACK option is closely related to the current BER. It is hard to quantify but SACK has not been very important on links of very high quality. However, the option will lead to higher efficiency when BER approaches 10^{-7}. In that case SACK can improve performance by more than 20%. This percentage goes down if the link becomes worse. During all experiments, using SACK has never had negative impact on the quality of service.

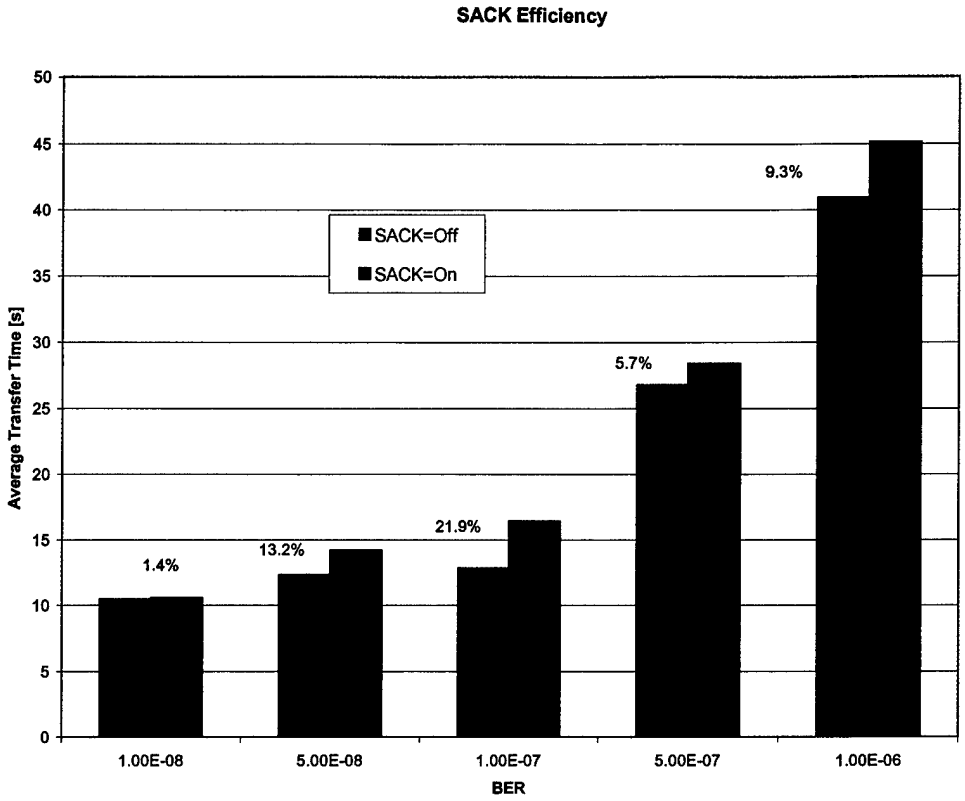

FIG. 6.5: SACK EFFICIENCY

6.5 SKYNET Experiments

The delay and the probability of errors in a circuit limit TCP's performance. These limitations become more important as new satellite systems offer much higher data transmission rates than those available in the past, but still are constrained by long path delays.

Within this study a testbed was established using a Data Link Simulator. It was shown that increasing the size of TCP's window and using a selective acknowledgement scheme lead to higher efficiency. To compare the results from the simulation with actual satellite links, we had the opportunity to run experiments on the SKYNET 4D satellite.

A 1024kbps circuit was established with modems[12] running QPSK and FEC coding. In general, the connection was very reliable but some burst errors were observed. BER was approximately $<10^{-6}$ but could not be measured accurately while transferring data. We

[12] ComStream 701

did not use interleaver techniques to mitigate burst error effects. RTT was measured to around 545ms. Fig. 6.6 shows the SKYNET experimental setup.

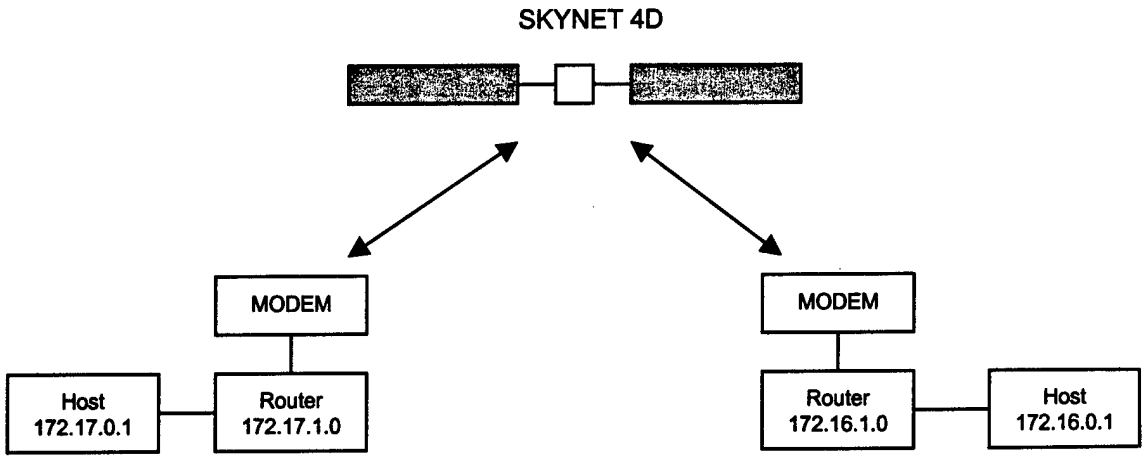

FIG. 6.6:SKYNET-EXPERIMENT

FTP was used on the application level. The TCP window was varied between 8 kbyte and 4 Mbyte. A 1 Mbyte file was transferred 50 times with and without using SACK for each window parameter.

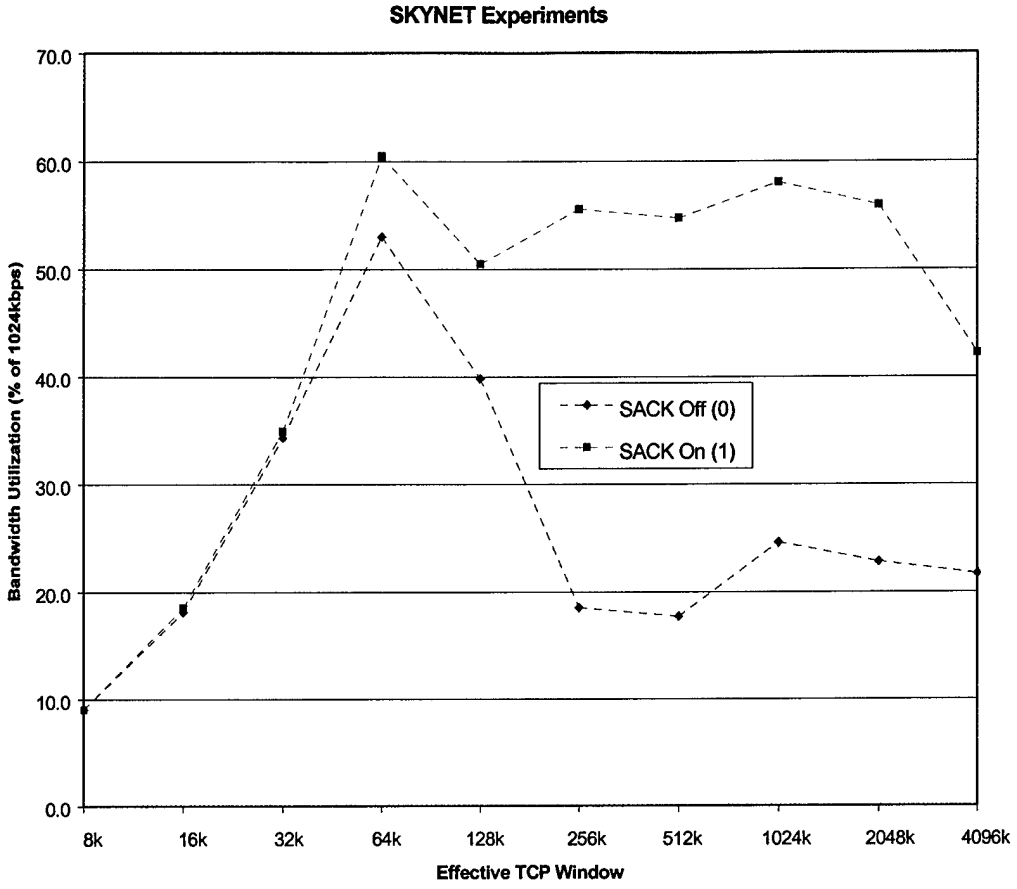

FIG. 6.7: BANDWIDTH UTILIZATION ON SKYNET CHANNEL

The results of the experiment are shown in Fig. 6.7. With the Windows NT standard of 8 kbyte TCP's flow control is not very efficient. Only 9% of the available bandwidth (1024kbps) was used by the TCP connection. Even with a larger window of 32 kbyte (the Linux default) utilization was only 34.9% with and 34.3% without SACK handling errors. The selection of a maximum window of 64 kbyte is in accordance with the bandwidth-delay product. Without using SACK an average level of 53% utilization was achieved with the 64 kbyte window. With SACK 60.4% of the bandwidth were used by the FTP connection. This has been the highest average value for an efficiency rate which has ever been observed during the experiments with the SKYNET satellite. Increasing the window size to values above the bandwidth-delay product had obviously no enhancing effect. Bandwidth utilization remained between 53% and 58%. A possible explanation for that was already outlined in section 6.3. As shown in Fig. 6.7 the transfer rate dropped

heavily when using larger windows without SACK. It is quite likely that burst errors were responsible for this behavior. Without SACK more RTOs occurred leading to timeouts and Slow Start phases. Because of this the bandwidth utilization was only around 20%. See ANNEX C for complete experimental results.

The result for the 4,096 kbyte window size showed a utilization value outside of what would be expected. While transferring data the measurement suffered from very bad conditions on that particular day. Burst-errors were observed more often than the days before. Even using SACK could not improve performance under these circumstances. Assuming similar conditions to the previous days a utilization of approximately 60% would be expected.

7 Conclusions and Recommendations

This study has examined the performance of TCP on satellite channels. As documented in the literature and observed in this report, the main reasons for poor throughput are the latency in satellite systems, high link error rates, and TCP's reaction to packet losses. Long delays directly affect end-to-end performance because of TCP's acknowledgement algorithm. Noise is most likely the reason for bit errors in wireless systems. However, TCP responds to all losses by invoking congestion control algorithms that reduce the value of TCP's flow control parameter (TCP window). Because of a smaller window size the transmission rate will be dropped significantly.

TCP options like "Window Scaling " and " Selective Acknowledgements (SACK)" were just introduced by the IETF to enhance performance and guarantee compatibility. Actual achievements were shown in this report. A testbed using a data link simulator was established and experiments according to different window sizes and SACK were conducted. Results were also verified by using a SKYNET satellite.

It was shown that using "Window Scaling" enhances performance. On a 1024 kbps satellite circuit, bandwidth utilization by one single FTP-connection became more than 60% while using the optimum window size (64 kbyte). With default values of Linux (32 kbyte) and Windows NT (8 kbyte) window sizes only 35% and 9% respectively were achieved. But managing flow control by determining the appropriate window size can be complicated, because maximum throughput is a function of the entire end-to-end connection. The factors responsible for the optimal window size are the bandwidth and the round trip latency or delay found on a circuit. The bandwidth-delay product represents the maximum amount of data a network can handle, and therefore represents the optimal TCP window size for that special circuit. Setting a value that is larger has no positive effect but might cause recovery problems and waste resources. Conversely, a window will not lead to efficient network utilization if it is smaller than the bandwidth-delay product.

It was shown that TCP's congestion control algorithms error handling becomes more efficient when using SACK. The advantage had been significant while transferring data over the SKYNET satellite using SACK and Window Scaling. In that case burst errors dropped bandwidth utilization from 55% while using SACK to only around 20% without using SACK.

The advantages of low error rate links can not be overstated. All of the enhancements described above perform better at lower BERs, 10^{-7} or better.

This study has focused on solutions on the transport level. Higher gain FECs (Turbo Codes) which work on the data link level of the open systems interconnection (OSI) model might also enhance performance by improving the link quality and they should be tested, as should approaches on the application level. Using multiple parallel TCP data connections to transfer a single file can improve FTP performance, for example.

It is recommended that Window Scaling and SACK should be considered for further investigation. TCP/IP should not only be examined with geosynchronous (GEO) but could also include low-earth-orbit (LEO) and medium-earth-orbit (MEO) constellations that use intersatellitelinks and handovers for world-wide coverage which might cause further limitations.

8 References

[1] M. Allman, et. al.: "Enhancing TCP over Satellite Channels using Standard Mechanisms (RFC 2488)"; Internet Engineering Task Force, , January 1999

[2] W. Stevens: "TCP/IP Illustrated Vol.I"; Addison-Wesley, 1994

[3] D. Comer: "Internetworking with TCP/IP Vol.I and II."; Prentice Hall, 1991 and 1994

[4] V. Jacobson, et. al.: "TCP Extensions for High Performance (RFC 1323)"; Internet Engineering Task Force, May 1992

[5] M. Mathis, et. al.: "TCP Selective Acknowledgement Options (RFC 2018)";Internet Engineering Task Force, October 1996

[6] W. Stevens: "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms (RFC 2001)"; Internet Engineering Task Force, January 1997

[7] M. Allman, et. al.: "TCP Congestion Control (RFC 2581)"; Internet Engineering Task Force, April 1999

[8] M. Allman, et. al.: "Increasing TCP's initial Window (RFC 2414)"; Internet Engineering Task Force, September 1998

[9] Description of the TCP version implemented in Linux (Red Hat) see directory: /usr/man/man4/tcp.4

[10] "Cisco 1600 Series Software Configuration Guide"; Cisco Systems, 1997; see also: http://www.cisco.com

[11] "Operating Manual SX/12 Data Link Simulator"; Adtech, 1999

[12] C. Partridge, et. al.: "TCP/IP Performance over Satellite links"; NASA/CR, 1999

ANNEX A: Tuning Operating Systems

Almost all of TCP's flow control services depend on the size of the window that is advertised by a recipient, since this value dictates the maximum amount of data that can be sent after the initial congestion window (cwnd) has fully opened. Because of that it becomes particularly important to determine the optimal window size as it was shown for the Linux operating system. More information about this topic and the opportunity to tune other operating systems can be found on the following web-pages:

Tuning Unix-Systems:
http://www.psc.edu/networking/perf_tune.html#Linux

Tuning Windows-Systems:
http://www.microsoft.com/TechNet/winnt/Winntas/technote/ImplemntIntegra/ntopt6.asp

ANNEX B: Router Configuration

To show current configuration enter *show running-config* command when in PriviledgedEXEC mode:

```
Montreal(Toronto)>enable
Password: Montreal (Toronto)
Montreal#show running-config
Building configuration...

Current configuration:
!
version 11.2
service timestamps debug uptime
service timestamps log uptime
service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname Montreal
!
enable secret 5 $1$6RfE$srH1/M7SQAMXVI2/xYz.D1
!
username Toronto password 7 02320B4904081B2E
ip subnet-zero
!
interface Ethernet0
 ip address 172.17.1.0 255.255.0.0
!
interface Serial0
 description Leased Line to Toronto
 ip unnumbered Ethernet0
 no fair-queue
!
ip classless
ip route 0.0.0.0 0.0.0.0 Serial0
ip route 0.0.0.0 0.0.0.0 Ethernet0
ip route 172.16.0.0 255.255.0.0 Serial0
ip http server
logging buffered 4096 debugging
snmp-server community public RO
!
line con 0
 exec-timeout 0 0
line vty 0 4
 password 7 1124160B03000E0D08
 login
!
end

Montreal#
```

ANNEX C: SKYNET Experiments (Results):

Bandwidth: 1024kbps

Effective Window	8 kByte		16 kByte		32 kByte	
SACK	Off	On	Off	On	Off	On
Average Speed bps	94439	94766	190060	194257	359634	366250
Deviation in %	2.4	2.1	6.0	4.1	14.6	15.0
Bandwidth Utilization in %	9.0	9.0	18.1	18.5	34.3	34.9

Effective Window	64 kByte		128 kByte		256 kByte	
SACK	Off	On	Off	On	Off	On
Average Speed bps	555784	633080	416894	530013	193696	581495
Deviation in %	29.6	8.1	31.8	32.1	20.5	33.9
Bandwidth Utilization in %	53.0	60.4	39.8	50.5	18.5	55.5

Effective Window	512 kByte		1024 kByte		2048 kByte	
SACK	Off	On	Off	On	Off	On
Average Speed bps	185783	573480	257574	607742	239266	586083
Deviation in %	30.9	43.6	29.2	23.0	35.5	32.5
Bandwidth Utilization in %	17.7	54.7	24.6	58.0	22.8	55.9

Effective Window	4096 kByte	
SACK	Off	On
Average Speed bps	227604	442569
Deviation in %	44.8	352.5
Bandwidth Utilization	37	19

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Ottawa
3701 Carling Ave
Ottawa, ON, K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Enhancing TCP Performance over Satellite Channels (U)

4. AUTHORS (Last name, first name, middle initial)

Kammermann, Maik, and Latour, Hugues (Capt)

5. DATE OF PUBLICATION (month and year of publication of document)

August 2000

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

55

6b. NO. OF REFS (total cited in document)

12

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

5CA14

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DREO Technical Report 2000-079

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

In general the Transmission Control Protocol (TCP) works well while establishing end-to-end connections for common Internet services. However, there are two problem areas associated with performance over satellite channels: propagation delay and channel noise effects.

Noise is a common problem in wireless technologies and consequently the bit-error-rate (BER) is significantly higher than in wired networks. TCP is particularly vulnerable to BER because it is a reliable protocol. TCP will retransmit lost or corrupted data when errors are detected. However, the main design goal was to avoid congestion-collapse in networks. The protocol has no means to decide whether a loss event was caused by congestion (buffer-overflow) or by corruption (noise, jamming). Nevertheless, TCP should react in a different way, depending on the type of errors: it should immediately retransmit outstanding data if the loss was caused by noise, and it should reduce network traffic if congestion was the reason for dropping data-packets.

Currently every loss indicates network congestion for the TCP standard version defined in RFC 0793. As a result it will reduce its sending rate significantly by invoking congestion control algorithms, regardless of the source of errors. TCP is a "Sliding Window" protocol that uses acknowledgements to verify proper data delivery. Such protocols allow the sender to transmit only a given amount of data before receiving an acknowledgement in return. After each acknowledgement the window size is increased and a larger number of segments will be transmitted. The time TCP needs to reach maximum throughput is a function of the time needed for delivery and acknowledgement of data. It is obvious that long delays will hurt performance on links characterized by a large bandwidth-delay-product. In this case TCP will poorly utilize the available bandwidth. Furthermore, long delays increase the amount of time TCP spends to recover from losses while throughput is already very limited during these periods.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Transmission Control Protocol/Internet Protocol
TCP/IP
satellite communications
TCP performance
IETF RFC 0793